Malware Loaders

Ronald E. Crane, J.D., B.S.C.S.

December 11th, 2005

Taxonomy

Vendor or a vendor's rogue employee.

Applicability

Any computer-based voting equipment, including DREs, DREs with VVPAT, ballot printers ("VVPB"), computer-based tabulators. etc. For brevity, this note concentrates on malware loaders in casting stations (DREs and the like).

Method

This attack allows the manipulation of vote totals, the alternation of ballots, or any other desired manipulation by facilitating the injection of malicious code into the voting application.

A malicious vendor (or a well-placed malicious employee of a vendor lacking sufficient internal controls and external supervision) places a small piece of code in its DREs' video BIOS[1], such that it will be invoked regularly during ordinary machine operation. This "malware loader" polls a communications device (such as a WiFi or WiMax port, broadband-over-powerline (BPL) port, IrDA port, Ethernet port, proprietary radio receiver, etc.) within the DRE for a signal to begin cheating. The vendor or malicious employee arranges to broadcast this signal during elections in which it/she wishes to cheat.

When the malware loader receives the cheating signal, it disables CPU interrupts to prevent interruptions from the operating system or any other applications. Then it locates a small area of unused memory and copies the "malware bootstrap" into it.[2] After doing

---

1 The code could be placed in the mainboard BIOS, in an FPGA or ASIC, or in the operating system. I have chosen the video BIOS because it illustrates the technique most clearly, and is more difficult to accomplish than placing the code in the operating system.

2 This exact arrangement assumes a single-tasking operating system. On a multitasking OS, the malware bootstrap would first have to find the process housing the voting application, probably by parsing the OS's process table. Then it would have to find an unused area within that process's address space.

so, it locates the address of a function called periodically by the voting application.[3] This function could do anything or nothing, just so long as the voting application calls it relatively frequently. For example, it could be a function that flushes audit records to persistent media, animates a logo, updates the time on the display, etc.

The malware loader modifies the first few instructions of the periodic function to contain a jump to the malware bootstrap, clears the CPU's instruction pipeline, re-enables interrupts, and returns control to whatever invoked it.

Eventually the voting application receives control, and eventually it calls the periodic function. When it does so, the periodic function jumps to the malware bootstrap, which knows all about the voting application and the communications device. It uses the communications device to load data and/or code ("cheating information") to make the voting application do anything the malicious vendor or employee desires. Once the application is compromised, it can even subject the machine to realtime remote monitoring and control, enabling the cheater to detect and evade parallel testing.

The malware bootstrap remains in memory to supervise further cheating, possibly including replacing the compromised voting application with the original application after the polls close, erasing itself, or copying the cheating information to "unused" areas of persistent storage for later use.

Note that this cheat works even if the entire voting application and operating system are publicly reviewed, found completely honest, and are properly and honestly loaded into the voting machines.

Resource Requirements

The cheater can be a vendor, a well-placed employee of a vendor lacking sufficient internal controls and/or external supervision, an integrator, or any other actor who can control the contents of the machines' firmware. Since firmware often is stored in dynamically-rewritable persistent memory (e.g., flash), a virus-writer, hacker, or anyone who can cause a program to be run on the machine to be compromised might also be able to emplace a malware loader.[4]

---

[3] The easiest approach is to use a function at a fixed address. By matching versioning information from the voting application (e.g., the fact that it printfs "Voting App. v.1.4.5" to the display on startup) with similar information from the communications device, a more advanced malware loader easily could look up the appropriate address for any voting application version. Of course this address has to reside in the voting application's address space; see note 2.

[4] The Chernobyl virus caused its victims' systems to "melt down" by erasing their flash-based BIOS firmware. http://vil.nai.com/vil/content/v_10300.htm. A more advanced virus could modify the firmware instead, perhaps emplacing a malware loader or other malicious software.

The cheater must be able to broadcast the cheating signal to the machines containing the malware loader, and must be able to follow it with the data and/or code that the malware loader expects.

Potential Gain

As many votes as the cheater wishes.

Likelihood of Detection

Very low. The VVSG do not require firmware inspections, and, even if they did, a malicious vendor simply would provide "honest" firmware to the inspectors, then ship machines containing malicious firmware. Since it is difficult, time-consuming, and expensive to inspect deployed machines' firmware, no one is likely to do so. Further, a crafty vendor could hide a malware loader in programmable logic, such as an FPGA or ASIC, that is also used to perform legitimate functions, such as controlling a video display. Such a loader is far more difficult to find than one hidden in an ordinary video or mainboard BIOS.[5]

Finally, since most elections are decided by small margins, and since exit polls have been subject to an extensive campaign aimed at discrediting them, it is unlikely that this cheat would be detected by monitoring election results.

Countermeasures

Preventative Measures

1. Prohibit all communications devices in voting machines. This approach, if well-enforced (it is difficult to enforce against a determined vendor[6]), makes it much more difficult remotely to monitor and control compromised machines. It does

---

[5] A crafty vendor might also consider using the capabilities provided by standard system-management and security firmware, such as that that supports Intel's Active Management Technology ("AMT"), http://www.intel.com/technology/manage/iamt/, to inject malicious code into voting machines. This approach cannot be detected by hardware inspections, since it does not modify off-the-shelf firmware. Instead it uses the very off-the-shelf firmware intended to help improve enterprise computer security to instead inject malware.

[6] Enforcement requires rigorous hardware inspections (i.e. rip to shreds) of a statistically-significant set of machines randomly chosen from the deployed base. See note 7. Further, communications devices are becoming smaller (and thus easier to hide) every day, and the trend will continue. Intel, for example, recently announced the development of single-chip WiFi. http://66.102.7.104/search?q=cache:PBxlV18sh3gJ:news.morningstar.com/news/DJ/M06/D17/200506170315DOWJONESDJONLINE000458.html+%22said+it+has+developed+prototype+chip+technology+that+can+handle+all+popular+forms+of+wireless+networking%22&hl=en.

not, however, prevent their compromise, since the cheating signal and following information can be loaded from ostensibly-unused areas of in-machine storage or from data cards used to record tabulated votes. The cheating information can even be loaded from image files and other data into which it has been steganographically encoded..

2. Require rigorous hardware inspections. This approach is a superset of (1). It involves regular random sampling of a statistically-significant set of deployed machines,[7] to inspect not only for hidden communications devices, but for malware loaders themselves.
3. Never run code from RAM. This approach makes it more difficult for the cheater to load the malware bootstrap and succeeding information into memory. It can, however, be worked around by loading code into RAM intended for data, or by housing more of the malware in the device housing the firmware.
4. Don't use electronic voting machines. This is the most secure approach. Hand-filled, hand-counted paper ballots are immune to this attack, and to many others affecting electronic voting machines.

Detection Measures

1. Parallel testing. Rigorously-conducted parallel testing of a statistically-significant set of randomly-selected machines should be able to detect the effects of some kinds of malware loaders.[8] Such testing probably will not be able to detect malware loaders that enable realtime monitoring and control of machines, since the cheaters may learn which machines are being tested, either directly (by knowing or observing the testing teams' schedules) or by observing the voting patterns on the entire machine base.
2. Voter verified paper ballots or paper trails. These measures enable the voter to detect the operation of malware loaders that transmute votes during casting: a significant advantage over unaided DREs. Note, however, that the proportion of voters who will verify their ballots or trails is unknown and is likely to decline over time, and the accuracy of their verification is unknown. Further, verification is not always meaningful. For example, a malware loader could generate marks on

---

[7] It is insufficient to sample machines that the vendor provides specifically for this purpose, since a determinedly malicious vendor will provide "honest" machines for this purpose, while deploying dishonest ones in the field. Similarly, it is insufficient to conduct the exam once, since an existing base of machines can be replaced, supplemented with new machines, or modified by firmware and/or hardware updates. And the set of machines sampled must be randomly chosen to prevent manipulation by the vendor or by others, and statistically-significant to ensure that it adequately represents the entire population of deployed machines. These procedures are lengthy, complex, expensive, and prone to shortcutting by vendors and elections officials. In consequence, they are likely to be ineffective unless experts from the general public have legally-enforceable rights and reasonable practical opportunities to supervise them. Even then, effective supervision may be spotty.

[8] As with all inspections involving voting machines, rigor is essential here. Any differences between the voting patterns that obtain during parallel testing and those during actual voting can be detected by sufficiently advanced malware.

voter-verified paper ballots that, while invisible to the voter, direct a cooperating tabulator to count her ballot differently from her intent. Or elections officials simply could fail properly to use a voter-verified paper trail. Finally, frauds that alter the presentation of the ballot to the voter (e.g., moving a disfavored candidate to the bottom of the ballot), or the manner in which her selections are accepted (e.g., making it more difficult to select a disfavored candidate), can influence the voter's actual selections, particularly if she is among the many voters who decide how to vote in the voting booth. Since these techniques create no mismatch between the vote the voter casts and the vote the machine records, their operation cannot be detected by voter verification.